Engineer's Notebook

A clear look at fuzzy PI control

By Peter Morgan

The second installment of *InTech's* advanced control tutorial series looks at one use of fuzzy logic in process control.

Fuzzy logic for industrial automation is gaining popularity, but how can it be applied to continuous process control? One way is with fuzzy PI control, a technique for adjusting the output from a proportional-plus-integral (PI) algorithm based on the results of an associated fuzzy logic algorithm.

Before considering fuzzy PI in detail, it is instructive to review the conventional PI algorithm in its incremental (or rate) form. This form of the algorithm is commonly used where the controller interfaces to a servo motor for final element positioning. For purposes of this evaluation, it allows the characteristics of fuzzy PI to be more easily compared with conventional PI.

The proportional-plus-integral algorithm in its series form is:

$$mv = G\left[\varepsilon + \frac{1}{\tau} * \int \varepsilon \, dt\right]$$

where

mv = manipulated variable (controller output)

G = controller or proportional gain

τ = integral action time (sec)

ε = controller error = +/- (process variable - set point), with + for direct-acting loops and - for reverse action

Presuming final output manipulation is an integration either in the program or at the final element — for example, when the output is treated as velocity demand for a servo motor — it is necessary to differentiate the PI algorithm to retain PI-type adjustment of the manipulated variable. In other words,

$$\frac{dmv}{dt} = G\left[\frac{d\varepsilon}{dt} + \frac{\varepsilon}{\tau}\right]$$

If the control system samples at an interval of T seconds, then

$$\frac{\Delta mv}{T} = G\left[\frac{\Delta\varepsilon}{T} + \frac{\varepsilon}{\tau}\right]$$

where

Δmv = required change in manipulated variable at end of each sample interval

$\Delta\varepsilon$ = change in error between samples

Solving for the change in manipulated variable,

$$\Delta mv = G\left[\Delta\varepsilon + \frac{T}{\tau} * \varepsilon\right]$$

If, after every scan, the output changes by Δmv as defined, then the output adjustment will match that obtained from a conventional PI algorithm implemented in a continuous (analog) controller.

Position demand can be obtained by integrating Δmv through a simple accumulation:

$$mv = mv_{last} + \Delta mv$$

Since the conventional, or crisp, control algorithm determines the change in output from the error and change in error, the fuzzy algorithm should rely on input memberships for these same variables. Consider three categories, or grades, for the error and Δerror: large-positive (L_{POS}), OK, and large-negative (L_{NEG}). As few as two membership grades can specify the input variables, but three grades more clearly demonstrate the principle.

Ignoring for the moment how the input variables are to be treated quantitatively, we can establish a rule set from our understanding of the incremental form of the PI algorithm. The first rule might be:

IF {$E = L_{POS}$} AND {$\Delta E = L_{POS}$} THEN {$\Delta MV = L_{POS}$}

where E and ΔE are the scaled error and change in error, and ΔMV is the scaled change in output.

This is entirely consistent with the equation for the rate form of the PI controller. It is also consistent with a rule based on intuition. In other words, if the error is already large and is increasing, then act to increase the output a lot.

The second rule could be:

IF {$E = L_{POS}$} AND {$\Delta E = L_{NEG}$} THEN {ΔMV = OK}

In other words, if the error is large and positive, but changing to be less positive, then do nothing (much!). This too might have been derived from the rate form of the conventional algorithm in so far as ε and $\Delta\varepsilon$ would tend to cancel, resulting in little change in the output.

hand, the process loop has two comparable first-order lags in the control valve and the transmitter. Also, on average, the process-loop dead time using control valve C is significantly longer when reversing direction. With the latter type of process response, it is unlikely that high controller gain can be used in a conventional proportional-plus-integral control scheme.

10% steps

Figure 6 indicates that with significantly larger amplitude steps, the motion of control valve A was velocity limited; it followed a nearly straight line, indicating the actuator being filled with the positioner relay wide open. The response of control valve B was still asymmetrical, but noticeably faster. The response of control valve C was faster than in the 2% steps, but still had an exponential behavior. Due to its single-stage positioner, control valve C was still significantly slower than the other designs.

As the step size in the open-loop tests becomes progressively larger, the control engineer should consider whether the control system is actually capable of delivering such a large command change to the final control element.

100% steps

The 100% steps were conducted in a separate test with the pump operating and the same load valve position as in the previous tests. The sample rate in the latter test was 55 per second. Full-range tests are sometimes called *stroking-time* tests. In most applications, this data is relevant only for emergency conditions.

Figure 7 shows that, for these largest steps possible, control valve A became significantly slower than the others. The major factors causing the slow stroking are the larger volume and lower supply pressure of this style of diaphragm actuator. The relays in the positioner of control valve A also had smaller flow area than that of control valve B. If fast response is desired for very large steps, use of a volume booster between the positioner and actuator would be appropriate with control valve A. The booster action should be limited to large-amplitude input changes, using a needle valve bypass around the booster relay.

The response of all three control valves was velocity limited over part of the 100% stroke. Control valve B still had faster response than control valve C. This apparently was due to a larger limiting airflow area in the positioner.

It is essential to recognize the strong amplitude dependence of the response-time data. The user must distinguish between the small-amplitude and large-amplitude requirements of the process

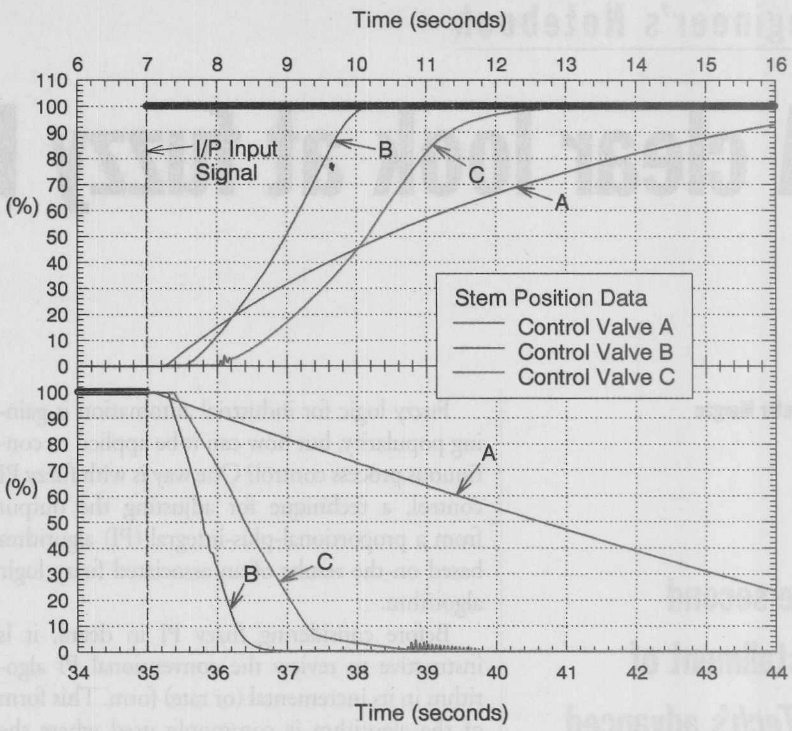

Figure 7. Stem-position response for 100% steps ("stroking-time" test)

control loop. Stroking-time data (Figure 7) cannot be scaled down to predict the small-amplitude behavior (Figure 5) of the control valve.

Try this at home

Users can easily apply step-response tests to obtain their own open-loop control valve performance data. They can measure dead band by making small moves about an operating point, and determine dynamic response. However, always use a step size appropriate to the control objective. Using steps larger than the disturbances the control loop will experience may result in selection of inappropriate equipment. Measurement of process variable changes, under process loading, is the ultimate confirmation of the desired behavior.

It is clear that broad performance claims, without considering all the components of the control valve, are not supported by the data. Superior performance requires superior design and integration of the valve, actuator, and positioner. For example, a belief that piston actuators always give better performance is clearly erroneous; combining a piston actuator with a low-performance positioner creates a slow-moving control valve. On the other hand, a low-friction valve coupled with a conventional diaphragm actuator and a two-stage positioner can give excellent dead band and small-amplitude response time. IT

Behind the byline

Mark T. Coughran is a research specialist at Fisher Controls International's Technical Center in Marshalltown, Iowa.

The complete rule set for the PI algorithm is shown in Figure 1, where for convenience the rules are shown on a grid (or matrix).

		Error		
		L_{POS}	OK	L_{NEG}
Delta Error	L_{POS}	L_{INC}	M_{INC}	OK
	OK	M_{INC}	OK	M_{DEC}
	L_{NEG}	OK	M_{DEC}	L_{DEC}

Figure 1. Fuzzy PI controller will change manipulated variable based on a "rule matrix."

Normalizing ε and Δε to bring E and ΔE within a range from -1 to +1 enables transformation of actual values to membership grades. The -1 to +1 range is used in most discussions of fuzzy sets, but, in practice, an alternative range may better suit the particular control system. For example, an integer value of 0-4,095 is more appropriate for a PLC that does not support floating-point math.

$$E = \frac{\varepsilon}{\varepsilon_{max}}$$

Assume raw error is in percent of transmitter range (a benefit when comparing scaling factor values with conventional gain and integral action settings). Then, requiring maximum incremental adjustment of the output when the raw error is 20% and change in raw error is only 1% per scan leads to ε_{max}=20 and $\Delta\varepsilon_{max}$=1. Since ε_{max} and $\Delta\varepsilon_{max}$ determine the degree of action for a given error and change in error, these scaling factors provide a means of tuning the algorithm. It will be shown later how these factors relate quantitatively to settings in the conventional algorithm.

As E, or ΔE, varies between -1 and +1 membership functions, such as those in Figure 2, determine the degree with which it qualifies in each of the membership categories.

In practice, the grades can be easily computed using simple algebra:

Grade(OK)	=	1-abs(E)	
Grade(L_{NEG})	=	abs(E)	for E less than or equal to zero
	=	0	for E greater than zero
Grade(L_{POS})	=	abs(E)	for E greater than or equal to zero
	=	0	for E less than zero

For illustrative purposes, assume control error is 5% of transmitter range and change in error per scan is 0.2%. If the error and Δerror maximum values are 30% and 0.5%, then E and ΔE are 0.167 and 0.4, respectively. The grades, or

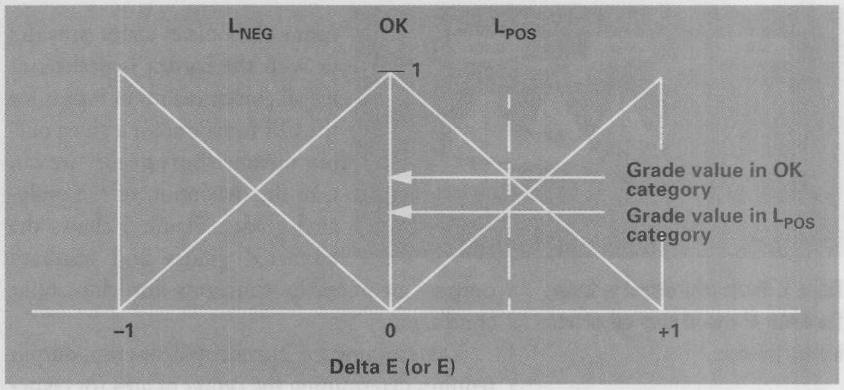

Figure 2. Fuzzy membership functions for inputs

degree of membership, of E and ΔE in the OK category are thus 0.833 and 0.6, respectively. Figure 3 shows how other grades follow from the application of the membership function equations for L_{POS} and L_{NEG}.

	L_{POS}	OK	L_{NEG}
E	0.167	0.833	0.0
ΔE	0.4	0.6	0.0

Figure 3. Membership grades for 5% control error and 0.2% change in control error per scan

From the rule matrix for fuzzy PI (Figure 1), when E is L_{POS} and ΔE is L_{POS} the change in output is required to be a large increase (L_{INC}); that E and ΔE are only partially L_{POS} means that the change in output for this rule must be only partially "large increase."

For Boolean variables, defined by the states 1 or 0, it is advantageous for manipulative purposes to consider the "AND" function as the multiplication of two, or more, logical states. In other words, 1×0 = 0, 1×1 = 1, 0×0 = 0. Taking the minimum Boolean state will also yield the desired logical result. So, too, in fuzzy logic is the minimum of the two input grades equivalent to the resultant grade for the consequent output.

Considering the grades for E and ΔE in the L_{POS} category (Figure 3) and the PI rule set (Figure 1), we would determine the output grade as 0.167 L_{INC}. Applying this principle to each PI rule results in output memberships shown in Figure 4.

	L_{POS}	OK	L_{NEG}
L_{POS}	L_{INC} 0.167	M_{INC} 0.4	OK 0.0
OK	M_{INC} 0.167	OK 0.6	M_{DEC} 0.0
L_{NEG}	OK 0.0	M_{DEC} 0.0	L_{DEC} 0.0

Figure 4. Moderate changes, M_{INC} and M_{DEC}, each have two grades, and no change has three grades in the output membership matrix.

Referring to the rule matrix (Figure 1), we may say, for instance, that the output will moderately increase if any of the antecedent rules call for a moderate increase; this is equivalent to an "OR" function. Noting that taking the maxi-

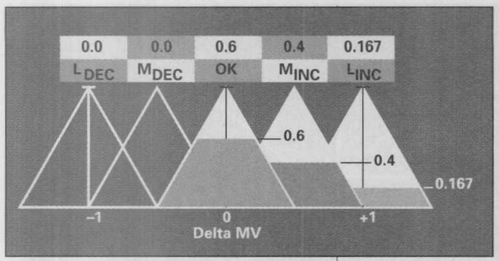

Figure 5. Obtain crisp output by finding the center of area of fuzzy output membership function.

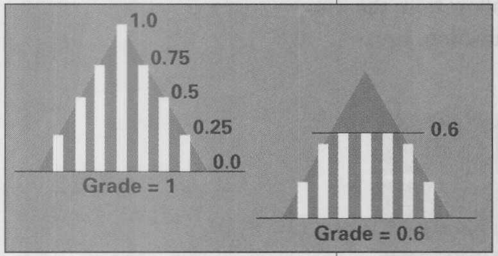

Figure 6. Representing output membership functions as series of discrete values simplifies calculation of crisp output.

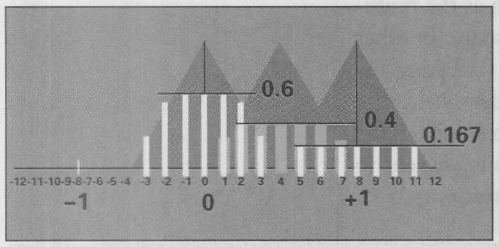

Figure 7. Discrete approximation to output membership functions map to output "axis" in process of determining crisp output.

mum of Boolean states provides us with the correct logical result for all combinations of inputs for the OR function, for a given output membership category we can take the maximum of the evaluated grades. Figure 5 shows the tabulated grades and resultant output membership functions for this input condition.

Determining the incremental, or crisp, output requires determining the center of area (or center of gravity) of the combined output membership functions. This is most easily accomplished by representing each membership function by a number of vertical elements. Figure 6 shows a single membership function represented by nine elements, including zero-value elements. Element value will equal the minimum of the value determined by the function's triangular envelope or the particular category's grade.

For each output category, element values may be mapped onto the output axis, as Figure 7 shows. In practice, this is accomplished by calculating V_i, the value of the function at the *i*th position on the output axis from:

$V_i = \mathrm{Max}\,\{F_1, F_2 \ldots F_J\}$

where F_1 through F_J are the membership function values applying at the *i*th position.

In this case, ignoring zero grades, only two membership functions apply at any position. At position +3, for instance, OK (value 0.25) and M_{INC} (value 0.4) apply; therefore, at position +3 the output function value V_i is 0.4.

Referring to Figure 7, if V_i is the value at the *i*th interval on the Δmv axis, then the center of area can be evaluated by dividing the sum of the individual moments about the zero axis by the sum of the individual values multiplied by the coefficient of normalization, 8 for this case:

$$\Delta MV = \frac{\sum_{-12}^{12} V_i \times i}{8 \sum_{-12}^{12} V_i}$$

For this example, the center of area, ΔMV, will be 0.327.

The actual change in output is the per unit change in output multiplied by the output scaling factor, or:

$\Delta mv = \Delta MV \times \Delta mv_{max}$

Thus, if the scaling factor, Δmv_{max}, is 2.5, when the algorithm output is 1, the change in final element position demand will be 2.5%. While it is possible to choose a scaling factor that matches the maximum rate of change in output to the velocity limit of the output actuator, this has an ameliorating effect on the response. For most purposes, it is better to use $\Delta mv_{max} = 100$ so that proportional-type action is unrestricted.

To obtain the final position demand, the calculated change in output is summed with the last output:

$mv = mv_{last} + \Delta mv$

For this example, with an output scaling factor of 100, the output change would be 32.7%.

The complete process, described schematically in Figure 8, continuously repeats this single-cycle computation (performed once per controller scan) with outputs calculated using the error and Δerror obtained at the beginning of each scan.

For study and prototyping purposes, a spreadsheet can easily implement this computational approach. Representing the axis of Figure 7 with a series of 25 spreadsheet cells, each cell will incorporate the formula for the membership function value V_i.

PI/fuzzy PI action comparisons

The first step in comparing the conventional and fuzzy PI characteristics is evaluating the

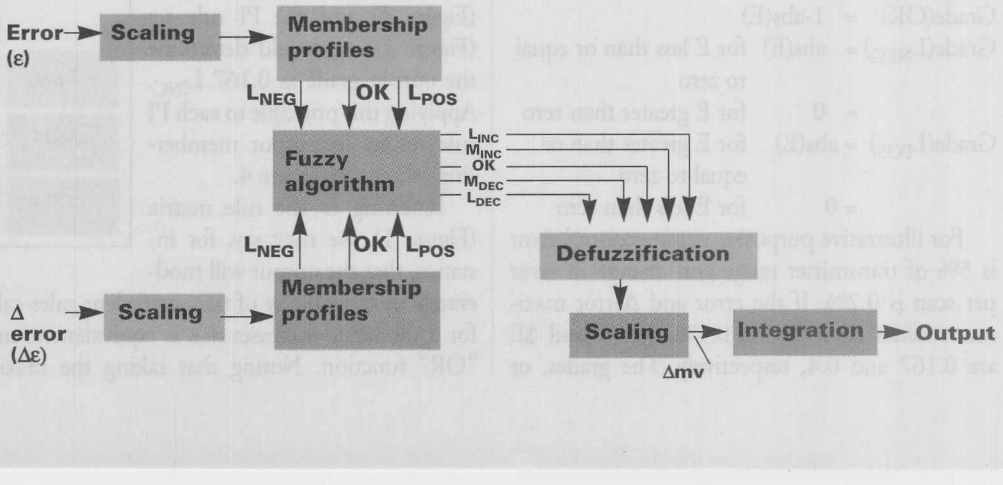

Figure 8. Fuzzy proportional plus integral

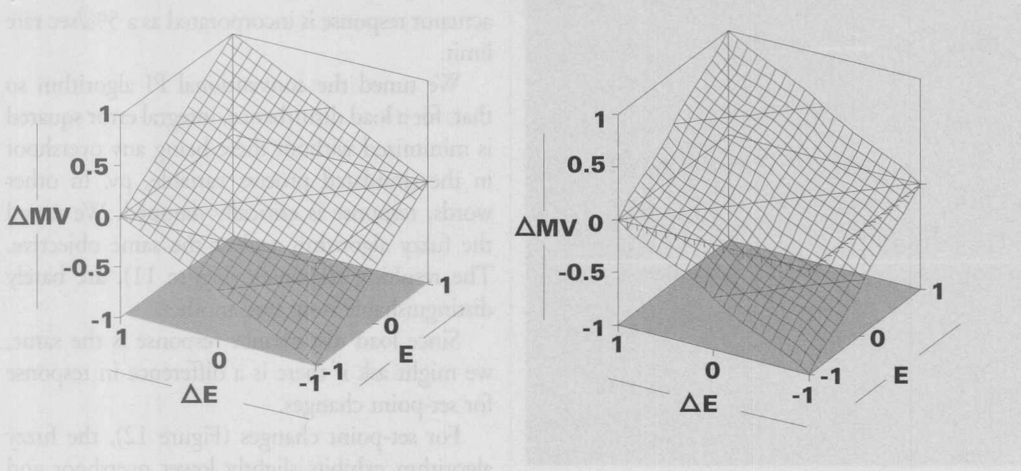

Figure 9. Conventional PI and fuzzy PI action planes. Fuzzy plane (right) is characterized by cusping introduced by the nonlinearity of the algorithm.

incremental output for the complete input range of inputs. Plotting incremental output on a vertical axis against error and Δerror on orthogonal horizontal axes yields a three-dimensional surface representing incremental output. Surface plots for both algorithms (Figure 9) show that, while the surfaces are similar, the fuzzy algorithm surface is characterized by cusping that results from the nonlinear relationship between change in output and error or Δerror. The ratio of fuzzy outputs to conventional outputs for each error condition, plotted in Figure 10, more clearly reveals the difference between actions.

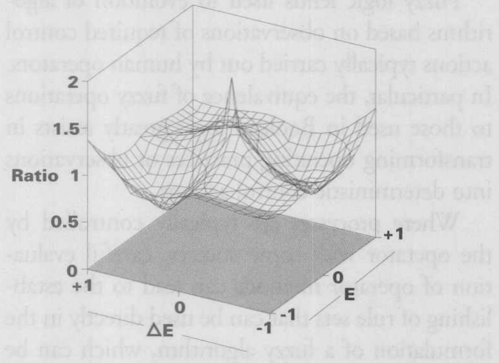

Figure 10. Plotting fuzzy output/conventional algorithm output more clearly illustrates difference between actions over input range.

From Figure 10, we may deduce that the fuzzy algorithm introduces nonlinearities that result in nonlinear dependencies in both proportional and integral actions. For small inputs, the fuzzy algorithm applies more aggressive correction than conventional PI. When Δerror has polarity opposite that of the error, i.e., when the process is moving in the right direction, the algorithm tends to be less aggressive.

PI/fuzzy PI parameter equivalence

While effective gain and integral action time for fuzzy PI control exhibit nonlinear dependence on error and change in error, it is possible to establish the approximate equivalence of fuzzy PI parameter settings to the gain and integral action settings for the conventional algorithm.

For the fuzzy algorithm, when error is zero, a change in error equal to the maximum value adopted in the scaling (normalization) process gives rise to a 50% change in output (for Δmv_{max} =100).

More generally:

$$\Delta mv = 0.5 \times \frac{\Delta mv_{max}}{\Delta \varepsilon_{max}} \times \Delta \varepsilon$$

Comparing this with the proportional component of the conventional PI algorithm in its incremental form, it follows that:

$$G = 0.5 \times \frac{\Delta mv_{max}}{\Delta \varepsilon_{max}}$$

Also, when the change in error is zero, an error equal to the maximum value adopted in the scaling process gives rise to a 50% change in output (for Δmv_{max} =100). In other words,

$$\Delta mv = 0.5 \times \frac{\Delta mv_{max}}{\varepsilon_{max}} \times \varepsilon$$

Referring to the integral term in the incremental form of the conventional algorithm, we may deduce that:

$$G \times \frac{T}{\tau} = 0.5 \times \frac{\Delta mv_{max}}{\varepsilon_{max}}$$

Rearranging and substituting for G, the equivalent integral action time can be obtained as:

$$\tau = T \times \frac{\varepsilon_{max}}{\Delta \varepsilon_{max}}$$

Tuning the fuzzy algorithm to meet the adopted response criteria for the process model used in this study requires setting $\Delta\varepsilon_{max}$ = 65 and ε_{max} = 2,500. These settings transform to an

Terminology

mv =	**controller output, i.e., manipulated variable**
e =	**controller error**
τ =	**integral time = 1/I, where I is the integral action rate**
G =	**controller or proportional gain**
$\Delta\varepsilon$=	**change in error during sampling**
T =	**sampling interval**
ε_{max}, $\Delta\varepsilon_{max}$=	**error and Δerror, which cause maximum incremental adjustment of output**
Δmv_{max} =	**maximum incremental adjustment of controller output**
E =	**scaled error =** $\frac{\varepsilon}{\varepsilon_{max}}$
ΔMV =	**scaled incremental change in manipulated variable =** $\frac{\Delta mv}{\Delta mv_{max}}$

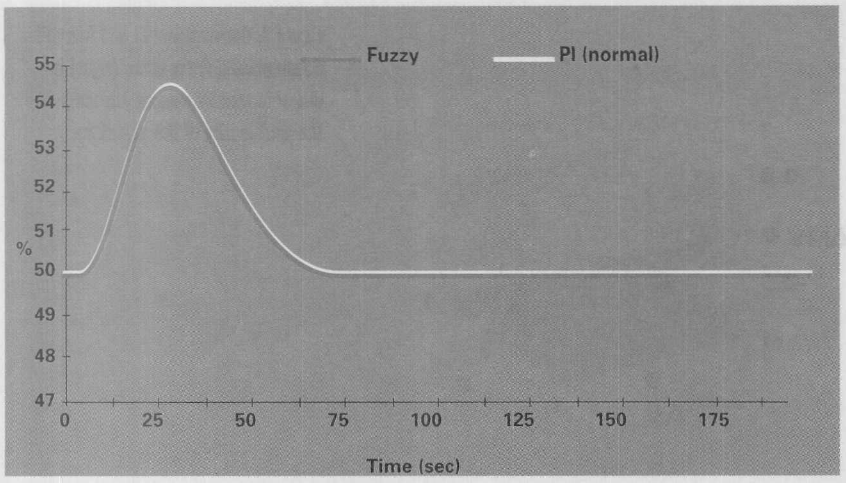

Figure 11. Fuzzy PI and conventional PI responses for 10% load disturbance are indistinguishable.

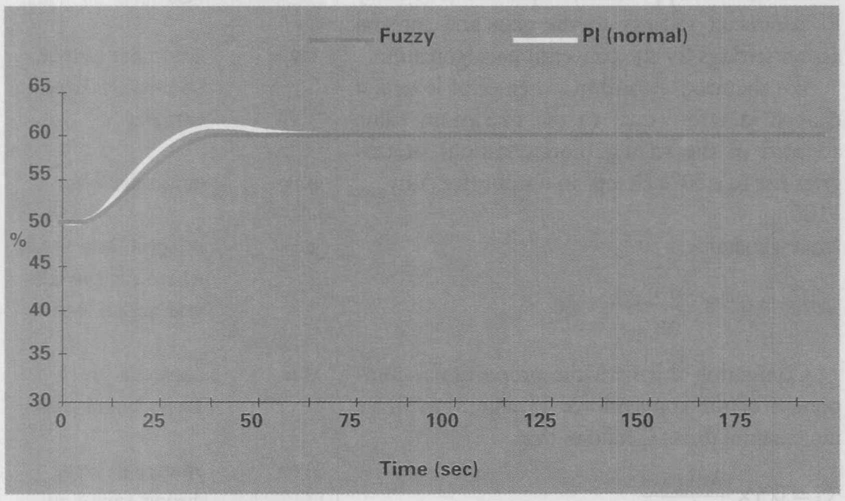

Figure 12. Fuzzy PI exhibits slightly less overshoot than conventional PI for 10% set-point change with controller optimized for load disturbance.

equivalent gain of 0.8 and integral action time of 19.2 seconds for a sample time of 0.5 seconds. The conventional PI controller requires gain of 1.25 and integral action time of 19.25 seconds. In other words, to obtain similar response, integral action times closely correspond, but the gain for conventional PI is somewhat higher than the transformed value. This is consistent with the demonstrated nonlinearity of the fuzzy algorithm and the fact that controllers are tuned to meet response objectives for a 10% load disturbance.

Who's the better performer?

In comparing responses, we strived to establish a good reference response for conventional PI. Often, tuning the reference PI goes no further than applying approximate methods, such as Ziegler-Nichols. In other words, without fine-tuning the reference algorithms against specified criteria, definitive response comparisons cannot be made.

The process model used in this analysis comprises a dead time approximation based on three 3-second lags and a 20-second first-order lag; actuator response is incorporated as a 5%/sec rate limit.

We tuned the conventional PI algorithm so that, for a load disturbance, integral error squared is minimized without there being any overshoot in the simulated process variable, pv. In other words, response is critically damped. We tuned the fuzzy algorithm against the same objective. The resulting responses (Figure 11), are barely distinguishable from one another.

Since load disturbance response is the same, we might ask if there is a difference in response for set-point changes.

For set-point changes (Figure 12), the fuzzy algorithm exhibits slightly lower overshoot and less tendency for overshoot to increase as set-point changes become smaller. However, comparing performance indexes, evaluated as integral-error squared, shows the conventional algorithm does somewhat better. We may quite fairly conclude that, for PI-type algorithms, there is no substantive difference in performance between the conventional algorithm and fuzzy algorithm.

Then why fuzzy logic?

While fuzzy logic may be no better than the conventional PI algorithm, it has comparable performance and can have its place.

Fuzzy logic lends itself to evolution of algorithms based on observations of required control actions typically carried out by human operators. In particular, the equivalence of fuzzy operations to those used in Boolean logic greatly assists in transforming operating practices or observations into deterministic control actions.

Where processes are typically controlled by the operator with some success, careful evaluation of operator methods can lead to the establishing of rule sets that can be used directly in the formulation of a fuzzy algorithm, which can be applied to control the process, reduce the requirement for constant operator intervention, and bring about performance improvements through the consistent application of the best operating practice. IT

Behind the byline

Peter Morgan is an engineering associate functioning as an internal consultant in process control with Syncrude Canada, Ltd., Fort McMurray, Alberta Canada. Mr. Morgan may be reached by e-mail at peter.morgan@syncrude. com.